EXHIBIT 11

Filed Under Seal

OPENING EXPERT REPORT OF SAMRAT BHATTACHARJEE REGARDING INVALIDITY OF U.S. PATENT NOS. 10,779,033 AND 9,967,615 AND OTHER ISSUES code by December 1, 2011 capture (although the feature was disclosed in the YouTube Remote patents well before Sonos's invention date, as I discuss in greater detail below).

162. Code for the YouTube Remote Application can generally be found in google3/java/com/google/android/apps/ytlounge/. The main application base class is defined in YtRemoteApplication.java.[3] The onCreate() function (line 232), which is executed when the application is started, creates a cloudService variable, of type CloudService. The cloudService object is an interface to the backend 'lounge server', which is set as "www.youtube.com," with URL path "/api/lounge" in the file C.java, lines 42-44. Incoming messages to the YTRemote application are processed by backend/CloudServerMessageListener.java (processMessage method, line 38).[4]

163. The YT Remote application communicates with the YouTube Lounge server using the BrowserChannel protocol. Code for the Lounge server can generally be found in google3/java/com/google/youtube/lounge.

164. The code for the Screen portion of the YTRemote system is found in the google3/flash/as3/com/google/youtube/modules/leanback directory.

2. Playback Queues And Recommended Videos

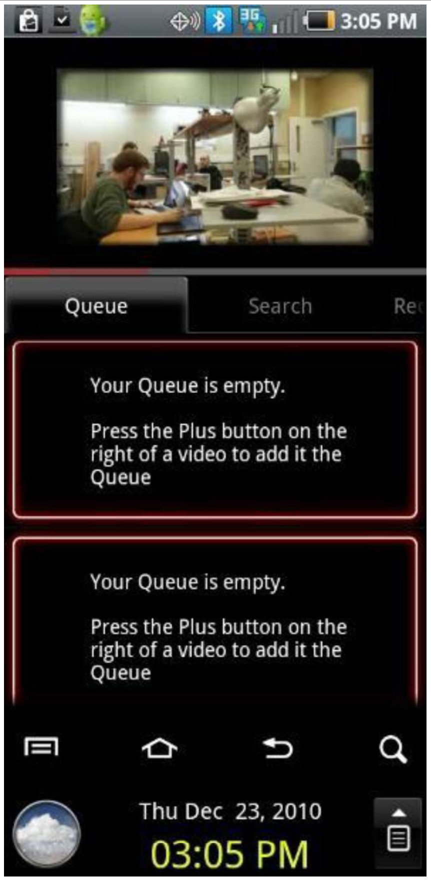
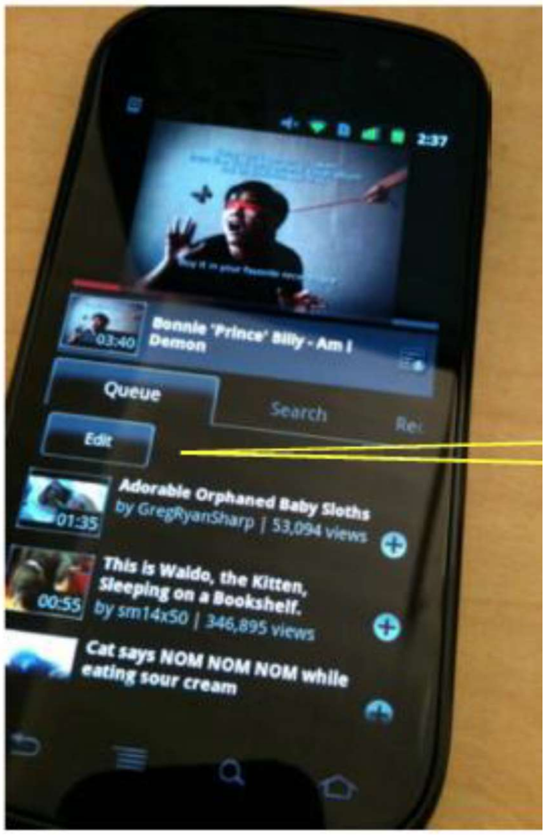
165. The YTR application allowed users to create queues of videos. Users could also play back lists of recommended videos provided by a YT cloud server.

166. For example, the image on the left below shows a YTR application prompting a user to "press the plus button on the right of a video to add it to the queue."

[3] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[4] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

https://web.archive.org/web/20101227075329/https://androidcommunity.com/youtube-remote-for-android-20101223/. *see also* **Video #3** at 0:34-0:50 (showing user adding items to the queue). Users could also edit the queue by, for instance, removing items from the queue or rearranging their order. The image on the right is from a presentation entitled "YouTube Leanback Remote Application" that is dated "April 2011" and shows an Android phone running a YouTube Remote application with a number of videos having been added to the queue. GOOG-SONOS-NDCA-00075619.

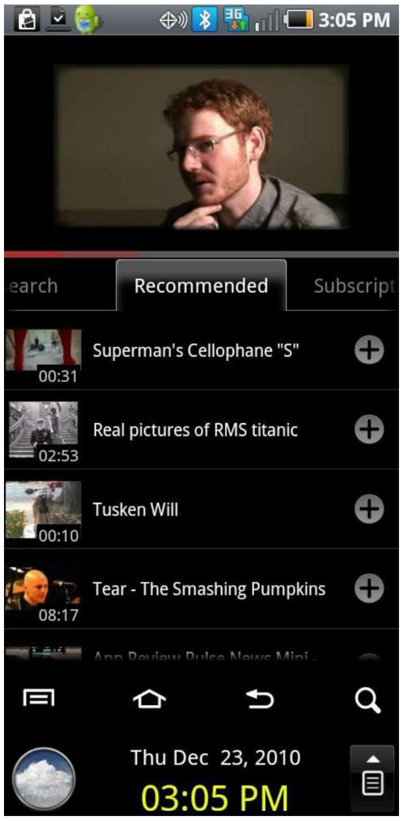
	

3:05 PM
Queue
Search
Re
Your Queue is empty.
Press the Plus button on the right of a video to add it the Queue
Your Queue is empty.
Press the Plus button on the right of a video to add it the Queue
Thu Dec 23, 2010
03:05 PM

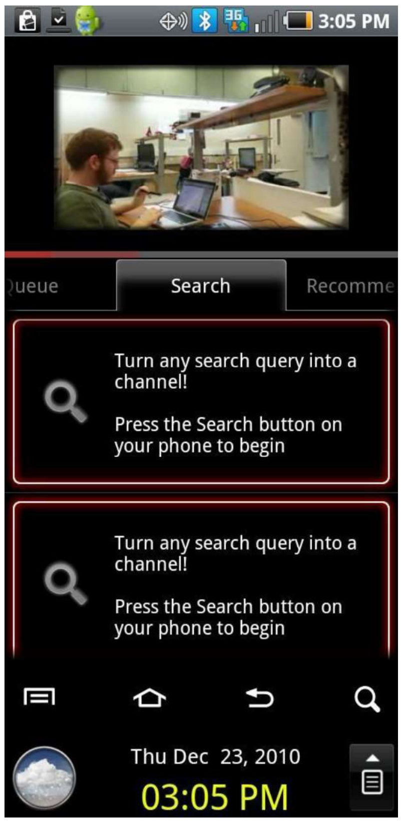
2:37
03:40
Bonnie 'Prince' Billy - Am I Demon
Queue
Search
Re
Edit
Adorable Orphaned Baby Sloths
by GregRyanSharp | 53,094 views
01:35
This is Waldo, the Kitten, Sleeping on a Bookshelf.
by sm14x50 | 346,895 views
00:55
Cat says NOM NOM NOM while eating sour cream

167. The YTR application also received server-driven lists of recommended videos. For instance, the YTR application received "recommended" videos from the YouTube servers, and could also retrieve a list of videos using the search function to create a channel. For example, the

YTR application display included a "Workspace" in which users could view various "stations" (sometimes also referred to as "channels"). These stations could be of various types, including user-editable queues, party queues (which I discuss in further detail in Section IX.A.2 below) "recommended" videos, and stations created using the YTR's search functionality. The images below show the Workspace and various stations:



Superman's Cellophane "S" 00:31
Real pictures of RMS titanic 02:53
Tusken Will 00:10
Tear - The Smashing Pumpkins 08:17

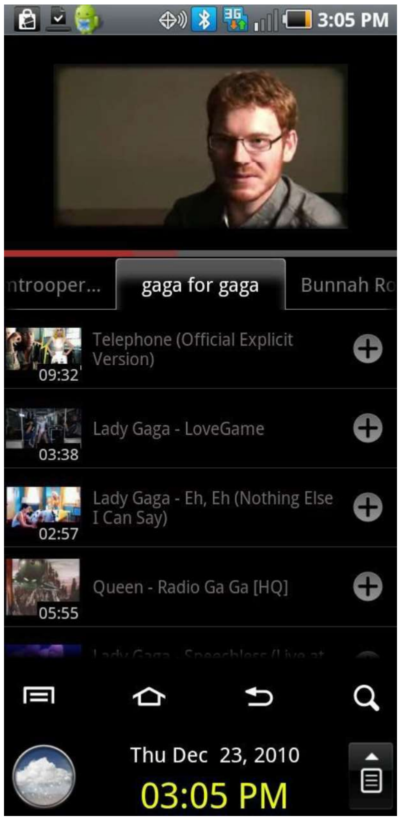


Turn any search query into a channel!

Press the Search button on your phone to begin

Turn any search query into a channel!

Press the Search button on your phone to begin



Telephone (Official Explicit Version) 09:32
Lady Gaga - LoveGame 03:38
Lady Gaga - Eh, Eh (Nothing Else I Can Say) 02:57
Queen - Radio Ga Ga [HQ] 05:55

https://web.archive.org/web/20101227075329/https://androidcommunity.com/youtube-remote-for-android-20101223/.

168. This functionality is also reflected in the July 12, 2011 capture of the YouTube Remote application.

169. For example, the source code confirms that the YTR application is able to add, remove, and update "stations" on its display (*see* WorkspaceManager.java[5]) and that these stations can be of various types, including user-editable queues, party queues, and "recommended" videos (*Id.*, lines 74-92). The method onUserProfileLoaded (ContentLoadingHelper.java[6], line 191) calls "loadAllstations" (*id.*, line 243) to add stations, including a list of "recommended" videos that are received from the server (*id.*, line 346). The backend/RealStationService.java[7] file creates the recommended station by retrieving a list of recommended videos from the YouTube cloud servers using a URL created by backend/station/UrlBuilder.recommended(). *See* RealStationService.java, line 149. The URL that is created to retrieve the list of recommended videos is found in UrlBuilder.java[8], line 177, and the host name in this URL is www.youtube.com, which is a cloud-based service. The recommended videos are returned as a list. *See* backend/PagedStationContentService.java[9], line 71; *see also* backend/logic/YouTubeService.java[10], lines 159-179 ("@return the list of videos in the station"); backend/station/GDataYouTubeService.java[11], lines 465-479. The code is configured to get 10

[5] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[6] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[7] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[8] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/station/

[9] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[10] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/logic/

[11] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/station/

recommended videos from the cloud service at a time. *See* PagedStationContentService.java[12], lines 16-39.

3. Party Mode And Party Queues Stored In The Cloud

170. After the initial release of the YTR application on November 9, 2010, Google added a feature to the YTR application called "party mode." This feature was reduced to practice in the July 12, 2011 capture of the YTR application, as I discuss below.

171. The party mode feature allowed users of the YTR application to share a playback queue with one or more of their friends. For instance, a host user (Alice) could create a playback queue containing YouTube videos. The host user (Alice) can then use her YTR application to invite one or more guests (e.g., Bob and Carol) to share her playback queue. The playback queue was editable by all members of the party (Alice, Bob and Carol) and could be played back on the mobile device of the host user and guest users, as well as the host user's Screen(s). The party queue was stored in a cloud server as a cloud queue, enabling multiple users to manage the queue during the party. When a host or guest user edited the party queue, the edit would be made in the cloud queue and then provided to the host and guest devices in the party.

172. In particular, a user running the YouTube Remote application on their mobile phone or tablet (the host user) could invite one or more guests to a "party." (*see* ContactListActivity.java[13], line 106 (inviteToParty), 197-208 (onActivityResult calls inviteToParty)). In party mode, the "host" user sends a queue of videos to a cloud server (e.g., a Lounge server) along with the list of guests. For instance, the inviteToParty function (*see*

[12] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[13] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

ContactListActivity.java, line 389) retrieves the current media queue (*see id.*, lines 419-423) and calls the partyModeManager.initPartyMode (*see id.*, line 424). The retrieved media queue can be any one of the stations I discussed above. For example, it can be a user created queue or a list of recommended videos (*see id.*, lines 419-423). Thereafter, the host YouTube Remote application goes into party mode (*see id.*, lines 428-430). The initPartyMode function (*see* RealPartyModeManager.java[14], lines 189-212) receives the video queue and guest name(s), and creates the new party queue. The queue on the host's mobile phone or tablet is then set to the party queue (*see id.*, lines 196) and the initPartyMode function then sends an INIT_PARTY_MODE message to the lounge server (*see id.*, lines 211). This message contains the video queue (*see id.*, lines 205), the current video (*id.*, line 207), the current time (*id.*, line 208), and guest name(s) (*id.*, line 210).

173. The Lounge server receives the queue sent by the host user and stores it as a party playlist (or party queue). For instance, the INIT_PARTY_MODE message sent by the host user is received by the YT Lounge server, and its handleMessage function. (*see* RealLoungeSessionManager.java[15] at line 326). The initPartyMode function (*id.*, line 776) sets the lounge session queue (the party queue) to be the queue received from the host (*id.*, lines 779-785). The sendInvitations function (*id.*, lines 946) sends a PARTY_INVITE message from the Lounge server to selected guest users for the party. This message contains the name and type of the party queue (*id.*, lines 795-796; *see also* RemoteQueueManager.java[16]).

[14] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[15] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/youtube/lounge/browserchannel/

[16] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

174. Guests (should they accept the party invitation) can receive the party queue that is stored in the cloud. For example, each guest user's YouTube Remote application receives the PARTY_INVITE message (CloudServerMessageListener.java[17], line 137) which causes the YTR application to issue a PARTY_INVITE intent. The intent contains the name of the host, the name of the shared party queue, and other information (*id.*, lines 145-158). This intent is serviced by BaseActivity.java (and a few other components) and calls onPartyInvite with the received host and station name. (*see* Baseactivity.java[18], lines 104, 125). The onPartyInvite function (*id.*, line 412) calls showInvite (*id.*, line 413). The showInvite function (*id.*, line 460) creates a dialog, and allows the invited guest user to accept or reject party invite (*id.*, lines 463-479). If the user accepts (*id.*, lines 480-491) the partyModeManager.joinParty function is called. The play queue is set as a new video play state (from remote queue), and the YouTube Remote application connects to the host's lounge session (*id.*, lines 486, 489).

175. Upon connecting to the Lounge server, the addDevice function is called on the Lounge Server (RealLoungeSessionManager.java[19], line 237) and causes the Lounge Server to send the cloud hosted party queue to the YouTube Remote application (*id.*, lines 260-264). The sendPartyQueueToDevice (*id.*, line 1065) calls sendPartyQueueToRemote (for the new remote YTRemote application that has just joined). The sendPartyQueueToRemote (*id.*, line 1074) sends a PARTY_PLAYLIST_MODIFIED message to the remote. This message contains the entire shared party queue (*id.*, line 1082). The guest YTRemote application receives the

[17] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[18] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/BaseActivity.java

[19] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/youtube/lounge/browserchannel/

PARTY PLAYLIST MODIFIED message (CloudServerMessageListener.java[20], line 96) with the shared party queue. Moreover, if one or more Screens are already connected to the lounge session at this point, then the sendPartyQueuetoScreen method is used to send the cloud hosted party queue to the one or more Screens. *See* RealLoungeSessionManager.java[21], lines 776-799. If a Screen joins the party at a later time, that Screen is also notified of the cloud hosted party queue. *Id.*, lines 1065-1072, 260-263.

176. The host and guest users can also continue to edit and manage the cloud-hosted party queue:

```
**
* Content service implementation for party mode. The definitive version of the
* playlist lives on the server, and multiple remote controls can change it at
* the same time.
*
* @author bobohalma@google.com (Ramona Bobohalma)
*/
ytremote/backend/SharedPlaylistContentService.java.
```

177. The file RemoteQueueManager.java manages the party queue.

```
/**
* Manages a remote queue (usually a party queue).
*
* @author danieldanciu@google.com (Daniel Danciu)
*/
```

[20] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[21] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/youtube/lounge/browserchannel/

RemoteQueueManager.java[22], lines 15-19. The party queue also supports queue management, for example adding and removing videos from the queue, reordering the queue, and clearing the queue. *Id.* at 20-83.

4. Registering Screens To A Lounge Session

178. A YTR application and one or more Screens are paired together by logging into the same YouTube account which registers them with the Lounge Server and adds them to a lounge "session." When in party mode, guests join the host user's Lounge session and are paired to the Screen(s) of the party host (if any).

179. A lounge session may have multiple devices associated with it, in particular a single lounge session may support multiple Leanback Screens (LoungeSession.java[23], lines 46, 204). *See also* LoungeSession.java, lines 21-23 ("A lounge session consists of several devices (remote controls and players) that communicate through the remote control server.").

5. Transitioning From Standalone Mode To Remote Control Mode

180. The prior art YTR application was able to transition from a first mode in which it was configured to play back videos on the mobile device, to a second mode in which it acted as a remote control for playback on the Screens. The first mode is sometimes referred to as "standalone" mode, and the second mode is sometimes called "remote" control mode. *See, e.g.*, WatchActivity.java[24], lines 1158-1196 ("Update the layout of the UI controls based on whether we are in landscape/portrait and remote/standalone mode....").

---

[22] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[23] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/youtube/lounge/model/

[24] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

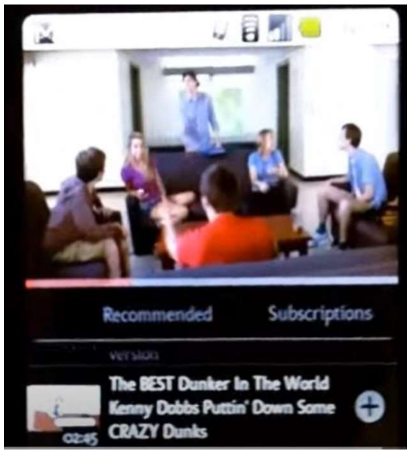
293. Initially, as I demonstrate later in this report, the written description of the '033 patent does not provide adequate support for this limitation. Thus, the disclosure of this limitation in the prior art exceeds what is provided in the written description.

294. I understand that this claim element includes one or more terms subject to construction by the Court. Specifically, I understand that the Court has construed the term "playback queue" to mean "[a] list of multimedia content selected for playback."

295. I further understand that the parties dispute the meaning of the term "remote playback queue." I understand that Google contends that the term means "a playback queue provided by a third-party application." However, I understand that Sonos disagrees with Google's interpretation and contends that the term encompasses a cloud queue or a list of recommended videos generated by a cloud server. *See* Section VI (Claim Construction). I agree with Google that a POSITA would understand the plain meaning of "remote playback queue" in view of the specification to refer to a playback queue provided by a third-party application.

296. In any event, I understand that claims are to be interpreted and applied in the same way for both infringement and invalidity. At least under Sonos's interpretation of the claims, the YouTube Remote system discloses and renders obvious this limitation.

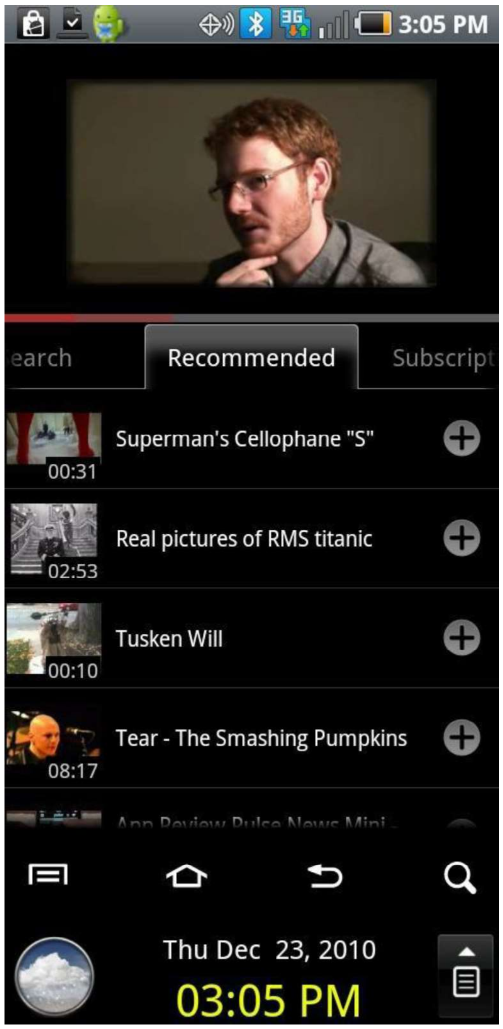
297. Google's YouTube Remote system allowed a user to operate in a standalone mode (first mode) in which the computing device (smartphone or tablet) was configured to playback videos locally on the computing device. For example, the image on the right from **Video #1** shows the user interface of the YTR application when a user is playing a video on his phone. A user of the YTR application could also rotate the phone, in

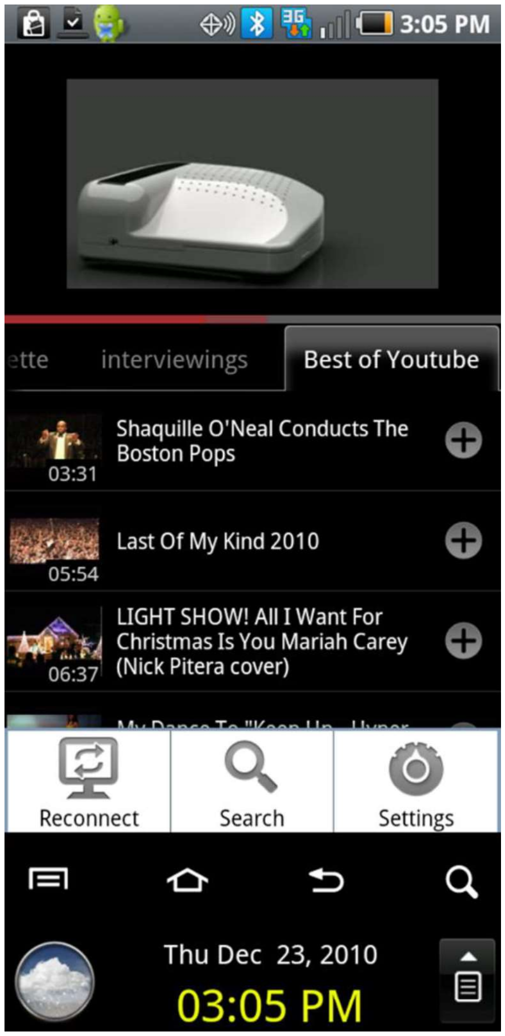

Recommended
Subscriptions
version
The BEST Dunker In The World Kenny Dobbs Puttin' Down Some CRAZY Dunks
02:45

which case the YTR application would play the video in full screen view. *See* **Video** #1 at 0:25-0:40. The user could also play, pause, fast forward or rewind the video using the graphical interface on the YTR application. *Id.* at 1:33-2:00.

298. When operating in standalone mode, the YouTube Remote application was also configured to be able to play back (i) a list of recommended videos that are received from a YouTube server in the cloud, and (ii) a cloud hosted "party queue." Under Sonos's interpretation, a YTR application is configured for "playback of a remote playback queue provided by a cloud-based computing system associated with a cloud-based media service" where it plays back a list of recommended videos provided by a cloud server or a cloud hosted party queue. *See* Section VI (Claim Construction), Section IX.A (YouTube Remote System). The prior art YTR application discloses this limitation under Sonos's interpretation.

299. First, the prior art YTR application satisfies this limitation when playing the list of recommended videos in standalone mode. The images below show examples of a user playing back a list of recommended videos using the YTR application on the mobile phone. The list of recommended videos are provided by a YT cloud server to the mobile device, as confirmed above in my analysis of the July 12, 2011 source code capture of the YTR application. *See* Section IX.A (YouTube Remote System).

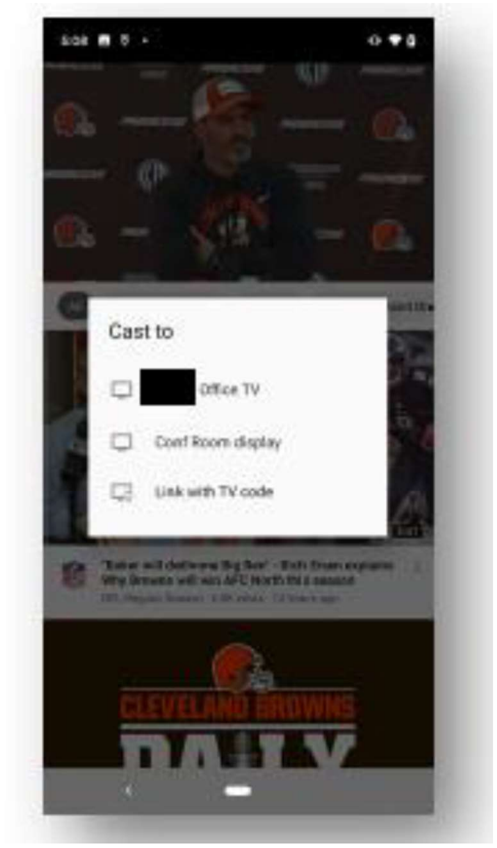

3:05 PM
earch
Recommended
Subscript
Superman's Cellophane "S"
00:31
Real pictures of RMS titanic
02:53
Tusken Will
00:10
Tear - The Smashing Pumpkins
08:17
App Review Pulse News Mini
Thu Dec 23, 2010
03:05 PM

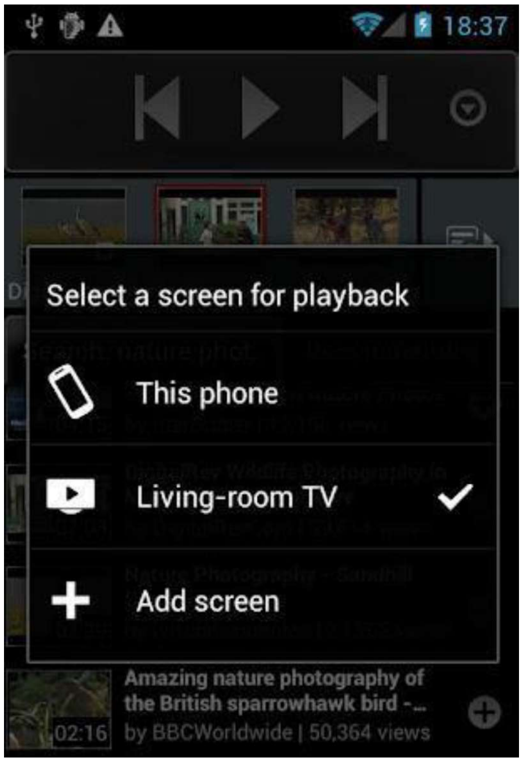


https://web.archive.org/web/20101227075329/https://androidcommunity.com/youtube-remote-for-android-20101223/

300. Second, the prior art YTR application satisfies this limitation when playing a "party queue" in standalone mode. As I explained in Section IX.A (YouTube Remote System), when party mode is initiated, the "host" user sends the queue to the cloud server (e.g., a Lounge server) along with the list of guests and, thereafter, the host YTR application goes into party mode and the playback queue sent by the host becomes a cloud hosted party queue. Guests (should they accept the party invitation) receive the cloud hosted party queue, and both the host and the guest can edit,

321. Second, Sonos's infringement contentions also point to the device-picker in the accused YouTube applications. *See* 1-20-2022 Sonos Supplemental Infringement Contentions, Ex. B ('033 Chart) at 19. On the left below is an excerpt from Sonos's contentions that illustrates the device-picker in the accused YouTube application, while on the right is the device-picker disclosed in the YouTube Remote Patent and implemented in the YouTube Remote system:




18:37
Select a screen for playback
This phone
Living-room TV
Add screen
Amazing nature photography of the British sparrowhawk bird -...
02:16
by BBCWorldwide | 50,364 views

The device-picker in the YTR prior art application satisfies these limitations in the same way Sonos contends the device-picker in the accused YouTube applications satisfies these limitations.

(vii) *Limitation 1.7: "based on receiving the user input, transmitting an instruction for the at least one given playback device to take over responsibility for playback of the remote playback queue from the computing device, wherein the instruction configures the at least one given playback device to (i) communicate with the cloud-based computing system in order to obtain data identifying a next one or*

> *more media items that are in the remote playback queue, (ii) use the obtained data to retrieve at least one media item in the remote playback queue from the cloud-based media service; and (iii) play back the retrieved at least one media item"*

322. Initially, as I demonstrate later in this report, the written description of the '033 patent does not provide adequate support for this limitation. Thus, the disclosure of this limitation in the prior art exceeds what is provided in the written description.

323. Sonos's applies this limitation broadly for purposes of its infringement allegations. For example, in its infringement contentions Sonos appears to accuse two separate instructions-one sent from the alleged "computing device" and another sent from a MDx server-as satisfying the limitation "transmitting an instruction for the at least one given playback device to take over responsibility for playback of the remote playback queue from the computing device." Specifically, Sonos claims that this limitation is satisfied by transmitting a setPlaylist message (the accused instruction) to a MDx server in the cloud, and then transmitting a separate message from the MDx server to the accused playback device. *See* 3-25-2022 Sonos Infringement Contentions, Ex. B ('033 Patent) at pp 27-50. I also understand that Sonos contends that the claim limitation does not require a single instruction that configures the at least one given playback device to perform sub-limitations (1)-(3), but rather that multiple instructions can collectively configure the at least one given playback device to perform sub-limitations (1)-(3).

324. Additionally, I understand that Sonos is reading the term "remote playback queue" to encompass a cloud-hosted queue and a list of recommended videos provided to a mobile device by a server in the cloud. *See* Section VI (Claim Construction).

325. I understand that claims are to be interpreted and applied in the same way for both infringement and invalidity. Thus, at least under Sonos's interpretation of the claims, the prior art

YTR system discloses and renders obvious this limitation when a user transfers playback of a party queue or a list of recommended videos. I discuss both scenarios below.

326. First, if a user is not in party mode and is playing back a list of recommended videos provided by a cloud server, a user input to transfer playback will result in the YouTube Remote application transmitting an instruction, namely a SET_PLAYLIST message, to the Lounge server. *See, e.g.*, RemotePlayerController.java[96], lines 92-102, LoungeStatusListener.java[97], lines 35-40, RealPlaystatePushService.java[98], lines 116-135, 264-280; Baseactivity.java[99], lines 230-239, 450-459. The Lounge server then relays the SET_PLAYLIST message to the Screens which will contain the list of recommended videos that were provided by the cloud servers. RealLoungeSessionManager.java,[100] lines 307 (handleMessage), 385 (call relayMessageWithinLounge), 851 (relayMessageWithinLounge function). The SET_PLAYLIST message is an instruction for the at least one given Screen to take over responsibility for playback of the list of recommended videos provided by the cloud servers, which, under Sonos's interpretation, is a "remote playback queue."

327. Second, this limitation is met when a user is in party mode and transfers playback of a party queue. When transferring playback of a party queue, the YTR application transmits an

[96] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[97] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[98] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

[99] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[100] In 2022-03-22_YTRemoteLeanbackAppsServer07122011/google3/java/com/google/youtube/lounge/browserchannel/

instruction for the at least one given playback device to take over responsibility for playback of the remote playback queue from the computing device. For example, upon receiving the user input to transfer playback from the YouTube Remote application on the mobile device to the one or more Screens, the YouTube Remote application will transmit a SET PARTY PLAYLIST and SET VIDEO messages to the Lounge server, as described above. *See* Section IX.A (YouTube Remote System). The SET PARTY PLAYLIST causes the Lounge server to transmit a SET PLAYLIST message to the Screens that contains the party queue (RealLoungeSessionManager.java[101], lines 679-688, 1088-1119), while the SET VIDEO message is relayed by the Lounge server to the Screens to commence playback of the party queue (id., line 385). The SET PARTY PLAYLIST and SET VIDEO are instructions for the at least one given playback device to take over responsibility for playback of the remote playback queue from the mobile device.

328. After receiving the SET PLAYLIST and SET VIDEO messages from the Lounge server in either of the above scenarios, the Screens are configured to communicate with the cloud-based computing system in order to obtain data identifying a next one or more media items that are in the remote playback queue. For example, for each item of media to be played back, the HTTP player issues a get video info request to obtain a Bandaid URL. *See* Section IX.A (YouTube Remote System).

329. The Screens then use the obtained data to retrieve at least one media item in the remote playback queue from the cloud-based media service and play back the retrieved at least one media item. For example, the data in the get video info response, including the Bandaid

[101] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/youtube/lounge/browserchannel/

URL, is used to retrieve the media content from the Bandaid CDN, which is then played back on the device. *See* Section X.A (YouTube Remote System).

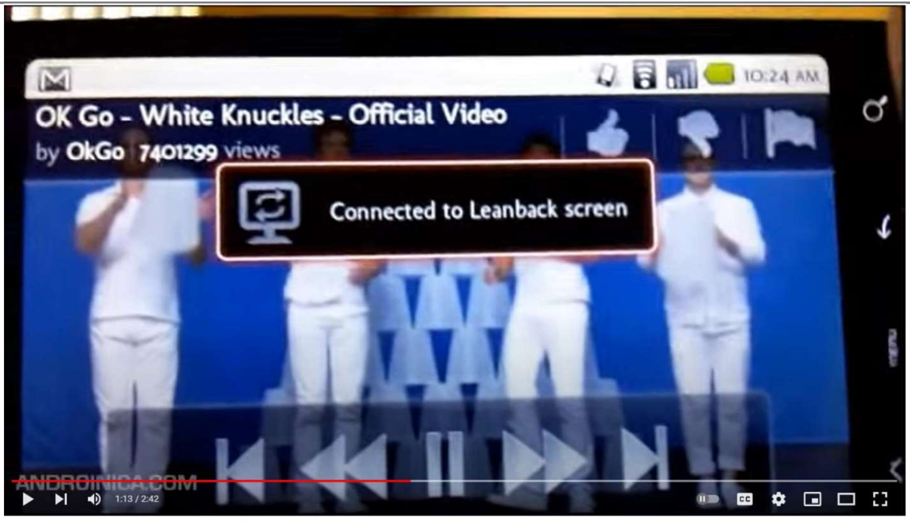
(viii) Limitation 1.8: "detecting an indication that playback responsibility for the remote playback queue has been successfully transferred from the computing device to the at least one given playback device"

330. Initially, as I demonstrate later in this report, the written description of the '033 patent does not provide adequate support for this limitation. Thus, the disclosure of this limitation in the prior art exceeds what is provided in the written description.

331. Moreover, Sonos applies this limitation broadly for purposes its infringement allegations. For example, Sonos's infringement contentions allege that this limitation is satisfied where the Cast button changes color after being selected. 11-21-2022 Sonos Infringement Contentions at 76-77 (stating that "Cast button" is "filled in" or turns "dark grey"). Sonos also appears to interpret "detecting an indication that playback responsibility for the remote playback queue has been successfully transferred" so broadly as to encompass a Cast-receiver retaining playback responsibility if transfer is unsuccessful. *Id.* at 84 ("Further confirming that each Cast-enabled display is programmed with the functional capability to detect an indication that playback responsibility for the remote playback queue has been successfully transferred from the Cast-enabled display to at least one Cast-enabled media player is the fact that the Cast-enabled display will take back over playback responsibility that was the subject of the "stream transfer" if the Cast-enabled display does not receive such an indication."). I understand that claims are to be interpreted and applied in the same way for both infringement and invalidity.

332. In my opinion, the prior art YTR system discloses this limitation. For example, after a user presses the Connect/Reconnect button to transfer playback to the one or more playback devices (Screens), the YTR application detects an indication that playback responsibility for the remote playback queue has been successfully transferred from the computing device to the at least

one given playback device and displays a "Connected to Leanback screen" dialog to inform the user that the playback responsibility has been successfully transferred. For example, the screenshot below form Video #1 shows a user pressing the Connect button to transfer playback to the playback device and, thereafter, the YTR application detecting successful transfer of playback responsibility and displaying the "Connected to Leanback screen" dialog:


10:24 AM
OK Go - White Knuckles - Official Video
by OkGo 7401299 views
Connected to Leanback screen
ANDROINICA.COM
1:13 / 2:42

How to control Google TV or YouTube Leanback with YouTube Remote

Video #1 at 1:06-1:19; *see also* LoungeStatusListener.java[102], lines 35-41; StatusBarNotifier.java[103], lines 63-79; strings.xml[104], line 138. This detection occurs after the YTR receives the SCREEN_CONNECTED message which triggers the BIG_SCREEN_CONNECTED intent. CloudServerMessageListener.java[105], lines 49-51. This

[102] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[103] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[104] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/ java/com/google/android/apps/ytlounge/res/values

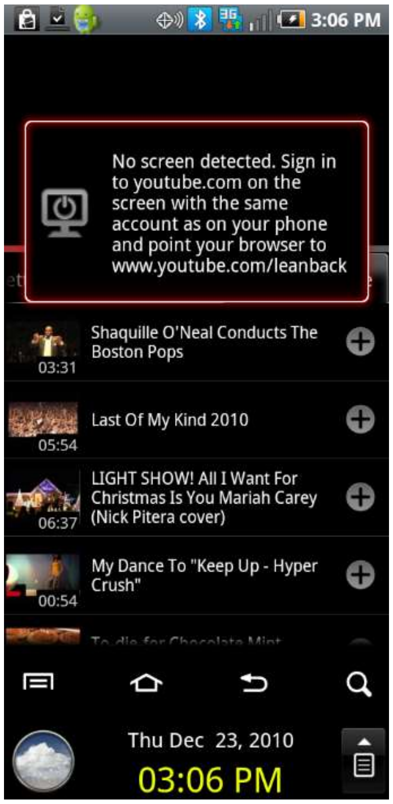
[105] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/backend/

intent also toggles the Connect/Reconnect button into a Disconnect button when playback responsibility has successfully been transferred. *See* BaseActivity.java[106], lines 74-77, 357-371, 506-537. Similarly, the YouTube Remote application detects playback has been transferred and updates the layout of the user interface to show the transport controls for controlling playback on the Screens, as can be seen in the image above which shows the YTR application with updated controls for controlling playback on the device. WatchActivity.java[107], lines 205-219, 1158-1196.

333. The YTR application will also retain playback responsibility if it does not detect that playback responsibility has been successfully transferred. For instance, if a user taps the Connect button and no Screens have been paired, the YTR application will not receive the SCREEN_CONNECTED. For instance, the image below shows the YTR application displaying a "No screen detected" indication. As just mentioned, above, Sonos has read this limitation broadly to encompass detecting an error that causes cause the computing device to retain playback.

[106] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

[107] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/


No screen detected. Sign in to youtube.com on the screen with the same account as on your phone and point your browser to www.youtube.com/leanback

3:06 PM
Shaquille O'Neal Conducts The Boston Pops
03:31
Last Of My Kind 2010
05:54
LIGHT SHOW! All I Want For Christmas Is You Mariah Carey (Nick Pitera cover)
06:37
My Dance To "Keep Up - Hyper Crush"
00:54
Thu Dec 23, 2010
03:06 PM

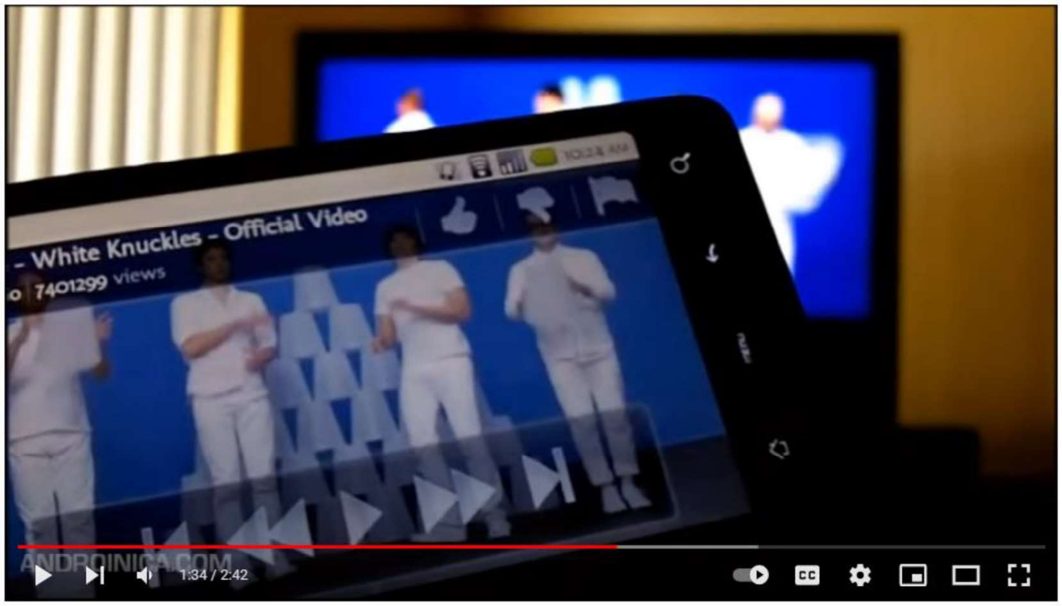
https://web.archive.org/web/20101227075329/https://androidcommunity.com/youtube-remote-for-android-20101223/

(ix) *Limitation 1.9: "after detecting the indication, transitioning from i) the first mode in which the computing device is configured for playback of the remote playback queue to ii) a second mode in which the computing device is configured to control the at least one given playback device's playback of the remote playback queue and the computing device is no longer configured for playback of the remote playback queue."*

334. Initially, as I demonstrate later in this report, the written description of the '033 patent does not provide adequate support for this limitation. Thus, the disclosure of this limitation in the prior art exceeds what is provided in the written description.

335. In my opinion, the prior art YT Remote system discloses and renders obvious this limitation.

336. After detecting an indication—as discussed in the prior limitation—the YTR application transitions from the standalone mode (the first mode in which the computing device is configured for playback of the remote playback queue) to a remote control mode (second mode). In the remote control mode the YTR Application is configured to control the one or more Screens and the YTR Application is no longer configured for playback of the remote playback queue. For example, Video#1, a screenshot of which is shown below, shows a user selecting the Connect button on the YTR application, which results in the user detecting an indication that playback transfer was successful an displaying the "connected to leanback screen") and also causes the YTR application to transition into the remote control mode where it no longer is locally playing the video and instead displays transports controls (e.g., pause, skip forward, skip backward, etc.) that the user employs to control playback on the Screen.



How to control Google TV or YouTube Leanback with YouTube Remote

Video #1 at 1:06-1:57 ("I can control YouTube Leanback directly from my phone"). While this video shows a user who has not initiated party mode, the July 12, 2011 source code shows that the transition from standalone to remote control mode is the same whether the user is in party mode or non-party mode. In particular, transferring playback from the YTR Application to the Screens causes the user interface to transition from controlling the local playback on the mobile device to controlling the Screens, and to stop playback on the mobile device. *See* WatchActivity.java[108], lines 207, 218; *see also id.*, lines 1158-1196 ("Update the layout of the UI controls based on whether we are in landscape/portrait and remote/standalone mode…."); Section IX.A. (YouTube Remote System).

337. To the extent Sonos argues that YTR application detects and indication that playback responsibility has been transferred concurrently—rather than prior to—with the transition to the second mode in which the transport controls of the YTR application are configured to control playback on the Screens, this limitation would have at least been obvious. A POSITA would understand that implementing the detection and updating of the UI transport control can be sequenced or done concurrently. Indeed, sequencing the operation so that the detection occurs before the UI transport control changes was obvious to try because it involved choosing from a finite number of identified, predicable solutions-namely, (1) modifying the transport controls prior to receiving the claimed indication, (2) modifying the transport controls concurrently with the claimed indication, and (3) updating the transport controls after the claimed indication-with a reasonable expectation of success. Sequencing the detection and the modification of the transports controls was also a straightforward design choice requiring only minimal modifications to software. Thus, it would have at least been obvious for the YTR application to sequence the operations such that it first detects an indication and displays the Connected to Leanback Screen

[108] In 2022-03-22 YTRemoteLeanbackAppsServer07122011/google3/java/com/google/android/apps/ytlounge/src/com/google/android/ytremote/

dialog, and then immediately thereafter modifies the transport controls so that they control playback on the Screens.

4. Claim 2

(i) *"The computing device of claim 1, wherein the instruction comprises an instruction for the cloud-based computing system associated with the media service to provide the data identifying the next one or more media items to the given playback device for use in retrieving the at least one media item from the cloud-based computing system associated with the cloud-based media service."*

338. In my opinion, the prior art YT Remote system discloses this limitations for the reasons I discussed above in connection with Limitation 1.7.

5. Claim 4

(i) *Limitation 4.1: "The computing device of claim 1, wherein the representation of the one or more playback devices comprises at least one selectable indicator for a group of playback devices that includes the given playback device and one or more other playback devices that are to be configured for synchronous playback of the remote playback queue"*

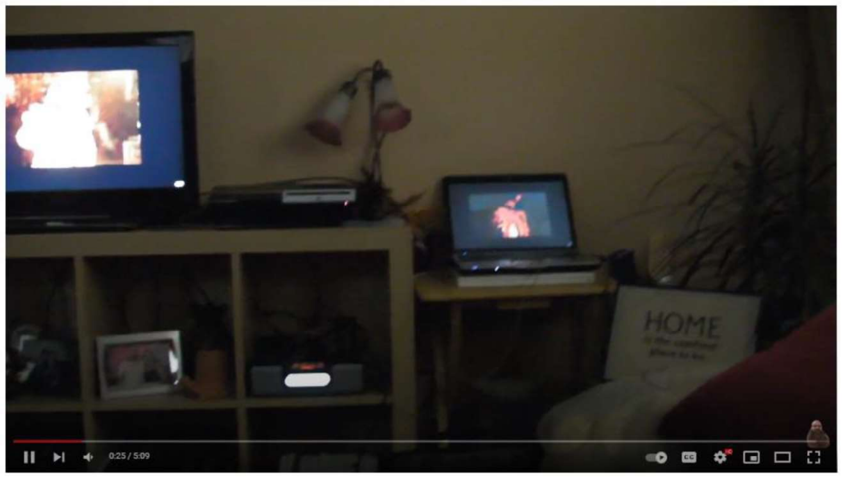
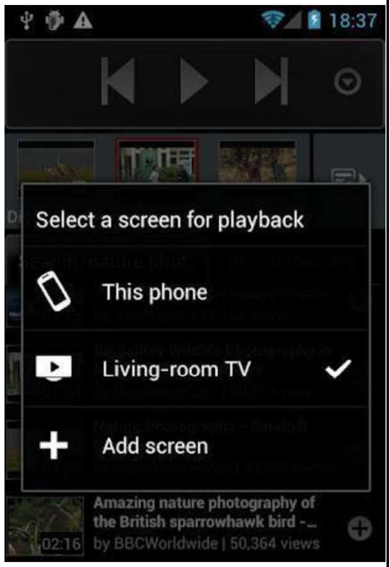
339. Sonos's applies this limitation broadly for purposes its infringement allegations. For example, Sonos's infringement contentions allege that this limitation is satisfied because the accused YouTube applications allegedly display an icon in the device-picker that when selected transfers playback to multiple playback devices, even though the accused YouTube applications do not allow users to select and group speakers together. *See* 1-20-2022 Sonos Infringement Contentions, Ex. B ('033 Patent) at pp 69-73. I understand that claims are to be interpreted and applied in the same way for both infringement and invalidity. Thus, at least under Sonos's interpretation the prior art YT Remote system discloses and renders obvious this limitation.

340. For example, I showed above that the YT Remote includes a representation of the one or more playback devices, such as a "Connect" button. Where two or more Screens have been connected to the Lounge session, the Connect button also comprises a selectable indicator for a

group of playback devices that includes the given playback device and one or more other playback devices that are to be configured for synchronous playback of the remote playback queue. Indeed, upon tapping the Connect button the video will transfer to playing back synchronously on the selected group of Screens.

341. This limitation is also rendered obvious in view of the device-picker in the YouTube Remote patent.

342. First, a POSITA would understand that an item in the device-picker can represent a group of playback devices. For instance, a Screen in the YTR system can comprise a group of playback devices, such as a computer that is connected via cable to a television, that play back in synchrony. The screenshot on the left below is from **Video #2**, which shows a user transfer playback from the YTR application to a PC that the user says is hooked up to his TV "via VGA" (0:22-0:45). Thus, a POSITA would understand that an icon on the device-picker (e.g., the Living-room TV icon on the right) can reflect a group of playback devices that will play in synchrony.

 Youtube (Lean back) Remote App for Android	 Select a screen for playback This phone Living-room TV ✓ Add screen Amazing nature photography of the British sparrowhawk bird -... 02:16 by BBCWorldwide | 50,364 views

Video #2 at 0:22-0:45 (left); https://web.archive.org/web/20120229040509/https://market.android.com/details?id=com.google.android.ytremote (GOOG-SONOS-NDCA-00108095 at 114). (right).

343. Second, the YTR patent discloses that the device-picker allows a user to select "one or more previously paired controlled devices." '998 patent at 10:67-11:6. A POSITA would understand this teaching could be implemented by having individual icons for each of the paired controlled devices, or by providing an icon for a group of previously paired controlled devices. At minimum, having the device-picker display an icon that represents a group of playback-devices was obvious. A POSITA would have recognized that allowing a user to display a single icon that represents a group of playback devices would be beneficial because it would allow a user to select and transfer playback to multiple devices with a click of a single icon, rather than having to select the icon for the multiple devices separately. Because the device-picker disclosed by the YTR patent already discloses that it can be used to select and transfer playback to multiple playback devices, representing those multiple playback devices using a single icon on the user interface would have been a straightforward and trivial implementation. The user interface would simply display a single icon that when pressed would function as if the individual icons of the group were selected. Indeed, by the time of the alleged invention of the '033 patent it was well-known that a group of playback devices could be presented on the display as a single icon for transferring playback. *See e.g.,* Section VIII.

6. Limitation 4.2: "wherein the user input indicating the selection of at least one given playback device from the one or more playback devices comprises user input indicating a selection of the group of playback devices."

344. In my opinion, the prior art YT Remote system discloses and renders obvious this limitation. *See* Limitation 4.1.

in retrieving the at least one media item from the cloud-based computing system associated with the cloud-based media service."

643. In my opinion, the prior art YT Remote system satisfies this limitation for the reasons I discussed above. *See* Limitation 1.7.

3. Claim 4

(i) Limitation 4.1: "The computing device of claim 1, wherein the representation of the one or more playback devices comprises at least one selectable indicator for a group of playback devices that includes the given playback device and one or more other playback devices that are to be configured for synchronous playback of the remote playback queue"

644. Sonos's applies this limitation broadly for purposes its infringement allegations. For example, Sonos's infringement contentions allege that this limitation is satisfied because the accused YouTube applications allegedly display an icon in the device-picker that when selected transfers playback to multiple playback devices, even though the accused YouTube Remote application do not allow users to select and group speakers together. *See* 1-20-2022 Sonos Infringement Contentions, Ex. B ('033 Patent) at pp 69-73. I understand that claims are to be interpreted and applied in the same way for both infringement and invalidity. At least under Sonos's interpretation, the prior art YT Remote system in view of the prior art Ramsay renders obvious this limitation.

645. As discussed above with respect to claim 1, the prior art Ramsay's existing device-picker permitted users to select multiple playback devices that are configured for synchronous playback of media in a queue by the time of the alleged invention. For instance, Ramsay discloses that its system uses "known discovery protocols" to discover playback device. Ramsay at 14:42-52. Those playback devices can be then be displayed on its device-picker. For instance, Figure 4 of Ramsay (below)- discloses a device-picker that allows the selection of a "speaker," a "group" of speakers, or a "zone".

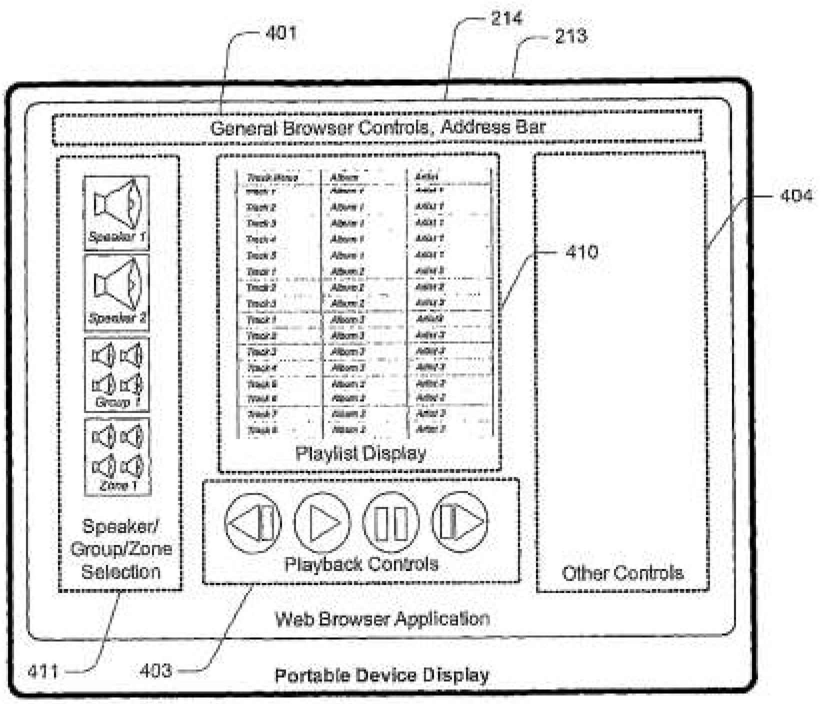


FIG. 4B

646. It would have been obvious to modify the YTR so that a selectable icon could be displayed. Because the YTR system already disclosed the ability to detect, display and transfer playback to multiple devices, allowing multiple devices to be represented by a single icon (rather than two separate icons) would have been an obvious design choice requiring only minor modification to the user interface display. A POSITA would have been motivated to make this modification because it was straightforward and routine, and because a user could benefit from being able to select a single icon to transfer playback to their devices, rather than having to individually select an icon for each device.

(ii) Limitation 4.2: "wherein the user input indicating the selection of at least one given playback device from the one or more playback devices comprises user input indicating a selection of the group of playback devices."

647. In my opinion, the prior art YT Remote in view of the Ramsay prior art renders obvious this limitation. *See* Limitation 4.1.